Master Theorem

The Master Theorem applies to recurrences of the following form:

$$T(n) = aT(n/b) + f(n)$$

where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function.

There are 3 cases:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a} \log^k n)$ with[1] $k \geq 0$, then $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ with $\epsilon > 0$, and $f(n)$ satisfies the regularity condition, then $T(n) = \Theta(f(n))$. Regularity condition: $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n.

Applied Master Theorem by this paper

Baris Ayyıldız 1901042252 3. Ayyıldız

1.a) $a=16$ $b=4$ $f(n)=n!$ $n^{\log_4 16} = n^2$ $n! > \Omega(n^2)$; case-3

Regularity Condition : $a f(\frac{n}{b}) \leq c f(n)$ for some constant $c < 1$ and all sufficiently large n

$$\frac{16 n!}{4!} \leq c\, n! \Rightarrow \boxed{c \leq \frac{2}{3}}$$

$T(n) \in \Theta(f(n)) = \Theta(n!)$ //

1.b) $n^{\log_2 2^{1/2}} = n^{1/4}$ $\log n < O(n^{1/4})$; case-1

$$\left(\lim_{n \to \infty} \frac{\log n}{\sqrt[4]{n}} = \frac{1}{\infty} \rightarrow \sqrt[4]{n} \text{ grows faster} \right)$$

$T(n) \in \Theta(\sqrt[4]{n})$ //

1.e) $n^{\log_3 3} = n$ $\sqrt{n} = O(n^{1-\varepsilon})$ for $\varepsilon > 0$, case-1

$T(n) \in \Theta(n)$ //

1.d / 1.f) These questions are not solveable by Master Theorem because $f(n)$ is an asymptotically negative function.

c) $n^{\log_2 8} = n^3$ $\qquad f(n) = 4n^3 = \Theta(n^3 \log^k n)$, $k \geq 0$

$\boxed{k=0}$

$T(n) = \Theta(n^3 \log n)$ //

g) $n^{\log_3 3} = n$ $\qquad f(n) = \dfrac{n}{\log n}$

Testing case-2 => $\dfrac{n}{\log n} \stackrel{?}{=} \Theta(n^3 \log^k n)$, $k \geq 0$

$\hookrightarrow \boxed{k=-1}$

Since $k < 0$ this recurrence relation is not solveable by Master Theorem //

2.a) $T(n) = 9T\left(\frac{n}{3}\right) + n^2$ $f(n) = \Theta(n^2 \log^k n)$, when $k=0$

$a = 9$
$b = 3$
$f(n) = n^2$

$\left.\begin{matrix} \\ \end{matrix}\right\} n^{\log_3 9} = n^2$ $\Rightarrow T(n) \in \Theta(n^2 \log n)$,,

2.b) $T(n) = 8T\left(\frac{n}{2}\right) + n^3$ $f(n) = \Theta(n^3 \log^k n)$, when $k=0$

$n^{\log_2 8} = n^3$ $\Rightarrow T(n) \in \Theta(n^3 \log n)$,,

2.c) $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$ $f(n) = \Theta(\sqrt{n} \log^k n)$, when $k=0$

$n^{\log_4 2} = \sqrt{n}$ $\Rightarrow T(n) \in \Theta(\sqrt{n} \log n)$

I'd choose the algorithm 'c'. Because it has the best time complexity with $\Theta(\sqrt{n} \log n)$

3.a) Merge sort works on the principle of divide and conquer. It first breaks down a list into sublists. Then it merges those sublists with each other. That's when the comparisons take place. If one sublist's all the elements are greater than the other it only make n comparisons (Here n is the size of that current sublist). In the worst case it's evenly distrubuted:

i.) $[\underbrace{1, 5, 3, 7}_{\text{5 and 7}}, \underbrace{2, 4, 6, 8}_{\text{6 and 8}}]$ number of comp : 17

ii.) $[1, 2, 3, 4, \underbrace{5, 6, 7, 8}_{\text{all the greatest numbers are in this side}}]$ number of comp : 12

3.b) Worst case occurs when the selected pivot element is the biggest or the smallest value in the list. If that's the case we can't divide the array into two even subarrays. And the best case happens when the selected pivot is the median value. Also worst case occurs if all the elements in the array are the same. If they are the same in every loop it will swap the items in the left and the right pointer until they meet.

CamScanner ile tarandı

i) [1, 1, 1, 1, 1, 1, 1, 1]

ii) [1, 3, 2, 7, 5, 6, 8, 4], least # of operations gives

1 3 2 7 5 6 8 4 pivot

1 3 2 $\boxed{4}$ 5 6 8 7 → divided into half

pivot

1 $\boxed{2}$ 3 $\boxed{4}$ 5 6 8 7

pivot

⋮

$\boxed{1}$ $\boxed{2}$ $\boxed{3}$ $\boxed{4}$ 5 6 $\boxed{7}$ 8 → divided into half

CamScanner ile tarandı

4) $T(n) = T(n/2) + C \rightarrow$ constant time operations

$T(n/2) = T(n/4) + C$

$T(n/4) = T(n/8) + C$

k steps

$T(1) = C$

$\boxed{n = 2^k}$

$n \rightarrow$ size

$k \in \mathbb{N}^+$

$n = 2^k$

$\boxed{k = \log n}$

$T(n) = T(n/4) + C + C$

$T(n) = T(n/8) + C + C + C$

$T(n) = \underbrace{C + C + \dots + C}_{k = \log n \text{ times}}$

constant

$A(n) = c \log n \in O(\log n)$ //

CamScanner ile tarandı

5.b) worst case: Quicksort analysis

$T(0) = T(1) = 0$, base case

$T(n) = n + T(n-1)$

$T(n-1) = (n-1) + T(n-2)$

$T(n-2) = (n-2) + T(n-3)$

$\vdots$

$T(n) = n + (n-1) + (n-2) + \dots + 2$

$T(n) = n^2 \dots \in O(n^2)$ //

Best Case:

$T(0) = T(1) = 0$, base case

$T(n) = 2T(n/2) + n$

=> From master theorem

$n^{\log_2 2} = n$ $\qquad f(n) = n = \Theta(n \log^k n)$, $k = 0$

$\Rightarrow T(n) \in \Theta(n \log n)$ //

Test function only requires $O(n)$ time. So we don't take it into consideration